Today we present various algebraic models of computation, and discover a few lower bounds.

1 Algebraic models of computation

1.1 Considered problems
Let \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) be a function that maps \( n \) elements of a ring \( \mathbb{R} \) into \( m \) elements of the same ring. Given \( x_1, x_2, \ldots, x_n \in \mathbb{R} \), compute \( f(x_1, x_2, \ldots, x_n) \).

Alternatively, for a function \( f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R} \), given \( x_1, x_2, \ldots, x_n \in \mathbb{R} \), determine \( y_1, y_2, \ldots, y_m \in \mathbb{R} \) such that \( f(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m) = 0 \).

1.2 Uniform model of computation
In the late 1980’s Blum, Shub and Smale came up with a uniform model of computation. It was a “Turing machine” over a ring. In this lecture we consider only non-uniform models of computation.

1.3 Algebraic circuits (Straight line programs)
An algebraic circuit is an acyclic network of gates with the following properties:

- the circuit has \( n \) inputs accepting \( x_1, x_2, \ldots, x_n \in \mathbb{R} \) and an arbitrary number of constants \( \alpha_i \) in \( \mathbb{R} \),
- the circuit has \( m \) outputs, \( y_1, y_2, \ldots, y_m \in \mathbb{R} \), and computes a function \( f \), i.e. \( f(x_1, \ldots, x_n) = (y_1, \ldots, y_m) \),
- each gate has two inputs and one output, and computes either the sum or product of input values (if \( \mathbb{R} \) is a field, we allow as well division).

The number of gates is a complexity measure of algebraic circuits.

Note that every algebraic circuit is equivalent to a straight line program in which every instruction corresponds to a single gate and has the form “\( v_i \leftarrow v_j \odot v_k \)”, where \( \odot \) is one of allowed operations.
1.4 Algebraic decision trees

An algebraic decision tree is a decision tree of the following properties:

- at each internal node we evaluate a polynomial of input elements $x_1, x_2, \ldots, x_n$, and branch, depending on whether the computed value of the polynomial equals 0 or not,
- each leaf contains a polynomial of the input elements which is the required values which we want to compute.

\[
\begin{align*}
& f_1(x_1, \ldots, x_n) = 0? \\
& f_2(x_1, \ldots, x_n) = 0? \\
& f_3(x_1, \ldots, x_n) = 0? \\
& \ldots \\
& p_1(x_1, \ldots, x_n) \\
& \ldots \\
& p_l(x_1, \ldots, x_n)
\end{align*}
\]

There are two complexity measures:

1. The depth of a tree.
2. The degree of polynomials at internal nodes.

1.5 Algebraic computation trees

An algebraic computation tree is a tree in which at each internal node we perform a single instruction of the form “$v_i \leftarrow v_j \diamond v_k$”, where $\diamond$ is one of basic operations allowed over $R$, and branch, depending on whether the computed value equals 0 or not.

\[
\begin{align*}
v_i \leftarrow v_j \diamond v_k & \\
\diamond & \in \{+, -, \times, /\}
\end{align*}
\]
2 Ostrowski’s conjecture

2.1 The problem: univariate polynomial evaluation

Given \( a_0, a_1, \ldots, a_n \in R \) and \( x \in R \), compute
\[
\sum_{i=0}^{n} a_i x^i.
\]

2.2 Horner’s rule

Horner’s rule enables us to evaluate a polynomial by \( n \) additions and \( n \) multiplications in the following way:

\[
\begin{align*}
 v_1 &\leftarrow a_n \cdot x + a_{n-1} \\
v_2 &\leftarrow v_1 \cdot x + a_{n-2} \\
&\vdots \\
v_i &\leftarrow v_{i-1} \cdot x + a_{n-i} \\
&\vdots \\
v_n &\leftarrow v_{n-1} \cdot x + a_0
\end{align*}
\]

2.3 The conjecture

Ostrowski came up in 1954 with the conjecture that Horner’s rule is optimal, i.e. one needs \( n \) additions and \( n \) multiplications (in the algebraic circuit model). He managed to prove that \( n \) additions are necessary, and in 1966 Pan proved that so are \( n \) multiplications.

2.4 Ostrowski’s lower bound

To show that we need \( n \) additions, we substitute \( x = 1 \), and the problem of evaluation of the polynomial reduces to the problem of computing the sum of coefficients.

Claim 1 To evaluate the sum of \( a_0 \) to \( a_n \) over a ring at least \( n \) additions are necessary in the algebraic circuit model.

Proof The proof goes by induction on \( n \). For \( n = 1 \), all that we can compute, not using additions, is \( ca_0 \cdot a_1 \), where \( c \in R \), which definitely differs from \( a_0 + a_1 \). For \( n > 1 \), the first addition in any straight line program looks like
\[
c_1 \prod_{i=1}^{n} a_i^{d_i} + c_2 \prod_{i=1}^{n} a_i^{e_i},
\]
and since it does not make sense to add constants as they can be hardcoded, we can assume that one of \( d_i \)'s or \( e_i \)'s is nonzero. Without loss of generality \( d_n \neq 0 \), for \( a_n = 0 \) the first addend disappears, and by the induction assumption we still need to spend \( n-1 \) additions to compute \( a_0 + a_1 + \cdots + a_{n-1} \). 

2.5 Pan’s lower bound

This time we substitute \( a_0 = 0 \). Note first that any algebraic circuit computes some polynomial in \( R[a_1, a_2, \ldots, a_n, x] \). A multiplication \( v_j \cdot v_k \) is insignificant if one of the following holds:

1. Both \( v_j \) and \( v_k \) belong to \( R[x] \).
2. One of $v_j$ and $v_k$ belongs to $R$.

Certainly, a multiplication that is not insignificant is significant. We will show that the number of significant multiplications is large enough in some more general case.

**Claim 2** Let $f : R^{n+1} \to R$ be a function of the form

$$f(a_1, a_2, \ldots, a_n, x) = \sum_{i=1}^{k} l_i(a_1, \ldots, a_n)x^i + r(x) + l_0(a_1, a_2, \ldots, a_n),$$

where each $l_i$ is a linear function, and $R$ is a field. An algebraic circuit computing $f$ has at least $\text{rank}\{l_1, l_2, \ldots, l_k\}$ significant multiplications.

**Proof** Look at the first significant multiplication. It has the following form:

$$\left(\sum_i c_i a_i + c_0(x)\right) \cdot \left(\sum_i d_i a_i + d_0(x)\right).$$

Without loss of generality $c_1 \neq 0$, and we restrict $(a_1, \ldots, a_n, x)$ so that the first term equals $c \in R$, achieving

$$c = \sum_i c_i a_i + c_0(x),$$

$$a_1 = \frac{c - c_0(x) - \sum_{i=2}^{k} c_i a_i}{c_1} = l(a_2, \ldots, a_n) + p(x)$$

for some linear function $l$ and polynomial $p$. Now we have a circuit that using one fewer significant multiplication computes

$$\sum_{i=1}^{k} l_i(l(a_2, \ldots, a_n) + p(x), a_2, \ldots, a_n) x^i + r(x) + l_0(l(a_2, \ldots, a_n) + p(x), a_2, \ldots, a_n)$$

$$= \sum_{i=1}^{k} l'_i(a_2, \ldots, a_n)x^i + r'(x) + l'_0(a_2, \ldots, a_n),$$

where $l'_i(a_2, \ldots, a_n) = l_i(l(a_2, \ldots, a_n), a_2, \ldots, a_n)$, and by basic linear algebra

$$\text{rank}\{l'_1, l'_2, \ldots, l'_n\} \geq \text{rank}\{l_1, l_2, \ldots, l_n\} - 1.$$

This implies by induction on the number of $a_i$'s that we need at least $\text{rank}\{l_1, l_2, \ldots, l_n\}$ significant multiplications. 

3 Fixed coefficients

If coefficients of the polynomial are fixed, that is we compute a function $f_{a_0 \ldots a_n} : R \to R$ such that

$$f_{a_0 \ldots a_n}(x) = \sum a_i x^i,$$

it turns out that we need at most $n/2 + 1$ multiplications, and that for most choices of coefficients this number of multiplications is necessary. The main idea is that we can express $f$ as

$$f(x) = q_1(x)(x^2 - b_1) + r_1(x),$$

there exists $b_1$ so that $r_1$ is of degree 0, and both $b_1$ and $r_1$ can be hardwired into a circuit. To show the lower bound we take $a_0, a_1, \ldots, a_n$ transcendental over $R$, and prove that if a program computes $\sum a_i x^i$ with $k$ multiplications, then $(a_1, \ldots, a_n)$ lie in a $2k$-dimensional extension of $R$. 

22-4
4 Evaluation in n points

Given $a_0, \ldots, a_n, x_0, \ldots, x_n$ in a field $K$, our goal is to compute $z_1$ to $z_n$ such that $z_i = \sum a_j x_j^i$. Using fast Fourier transform, we can achieve this in $O(n \log^{O(1)} n)$ time, and Strassen has proven that we need $\Omega(n \log n)$ operations in any algebraic computation tree. We will cover this topic in the next lecture.